

INF550. Conception et analyse d'algorithmes

Examen du vendredi 5 décembre 2008. 2 heures

Documents autorisés : poly, transparents du cours, énoncés de PC

Le barème est indicatif. On vous demande d'accorder une attention particulière à une rédaction soignée, qui sera un critère important dans l'évaluation.

Vous êtes embauché par une multinationale de renommée internationale pour inclure dans la prochaine version de son système d'exploitation une puissante suite logicielle d'aide à la pose de carreaux de salle de bain pour clients excentriques.

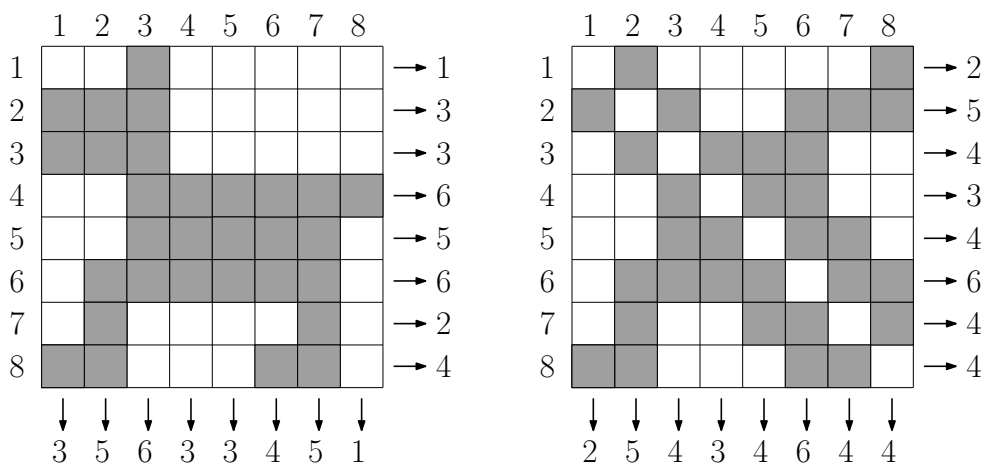


FIG. 1 – (a) un carrelage de taille 8×8 . (b) un carrelage symétrique de taille 8×8 .

I. Algorithme de flot maximum

Un client désire que soient posés des carreaux unité (1×1) blancs et noirs sur un mur de taille $n \times n$. Il souhaite de plus choisir le nombre de carrés noirs dans chaque ligne et dans chaque colonne. Vous devez proposer un carrelage qui satisfait ces conditions.

Il s'agit donc, étant donnés deux vecteurs de n entiers $\ell = (\ell_1, \dots, \ell_n)$ et $c = (c_1, \dots, c_n)$, de déterminer une matrice $M = (m_{i,j})_{1 \leq i,j \leq n}$ dont les entrées sont dans $\{0, 1\}$ et telle que $\sum_{i=1}^n m_{i,j} = c_j$ pour tout $j = 1, \dots, n$ et $\sum_{j=1}^n m_{i,j} = \ell_i$ pour tout $i = 1, \dots, n$. On suppose que $\ell_1 + \dots + \ell_n = c_1 + \dots + c_n$, sans quoi il n'y a évidemment pas de solution.

1. Utiliser la figure 1(a) pour donner un exemple de carrelage ayant comme vecteurs de projection $\ell = (1, 2, 3, 3, 4, 5, 6, 6)$ et $c = (1, 3, 3, 3, 4, 5, 5, 6)$.

Corrigé: *Permuter les lignes pour les mettre par ordre de ℓ_i croissants, puis faire de même avec les colonnes.* \square

2. Modéliser ce problème par un problème de flot maximum dans un graphe ayant $2n + 2$ sommets (source et puits compris). Quelle est la complexité d'un algorithme qui propose un carrelage par cette méthode ?

Corrigé: On considère le graphe formé d'une source s , d'un puits t et de sommets $x_1, \dots, x_n, y_1, \dots, y_n$, avec les arcs (s, x_i) de capacité ℓ_i , (x_i, y_j) de capacité 1 et (y_j, t) de capacité c_j pour tout i, j .

À un flot entier ϕ sur ce graphe on associe la matrice M donnée par $m_{i,j} = \phi((x_i, y_j))$: si le flot maximal est de valeur $\Phi = \sum_{i=1}^n c_i$ alors la matrice M donne un carrelage satisfaisant.

Inversement s'il existe une matrice $M = (m_{i,j})$ satisfaisante alors on construit un flot en posant $\phi(x_i, y_j) = m_{i,j}$ et en complétant par conservation du flot en les sommets : $\phi(s, x_i) = \sum_{j=1}^n \phi(x_i, y_j)$, $\phi(y_i, t) = \sum_{i=1}^n \phi(x_i, y_j)$. Ce flot respecte les capacités des arcs, il est de valeur $\sum_{i=1}^n c_i$, donc maximal.

L'algorithme de Ford Fulkerson trouve un flot maximal en temps $O(m\Phi)$, ici $O(n^3)$. \square

3. Soient $\ell^* = (\ell_1^*, \dots, \ell_n^*)$ et $c^* = (c_1^*, \dots, c_n^*)$ les vecteurs obtenus en réordonnant les ℓ_i et c_i en ordre croissant. Montrer, en utilisant le théorème FlotMax=CoupeMin qu'il existe un carrelage satisfaisant si et seulement si pour tout r et s avec $1 \leq r, s \leq n$ on a

$$\sum_{j=s+1}^n c_j^* - \sum_{i=1}^r \ell_i^* \leq rs.$$

Corrigé: Soit $C = C_1 \cup C_2$ une coupe avec $C_1 = \{s\} \cup X_1 \cup Y_1$ avec $X_1 \subset X = \{1, \dots, n\}$, $Y_1 \subset Y = \{1, \dots, n\}$, $C_2 = \{t\} \cup X \setminus X_1 \cup Y \setminus Y_1$.

La capacité de C est par définition $\sum_{i \in X_2} \ell_i + \sum_{i \in X_1, j \in Y_2} 1 + \sum_{j \in Y_1} c_j$. À $|X_1| = r$ et $|Y_2| = s$ fixés la coupe qui minimise la capacité est obtenue en prenant les r plus petits ℓ_i et les s plus petits c_j .

Pour qu'il existe un flot de flux $\Phi = \sum_{j=1}^n c_j$ il faut et il suffit donc que pour tout r, s , $\sum_{i=1}^r \ell_i^* + rs + \sum_{j=1}^s c_j^* \geq \sum_{j=1}^n c_j^*$ qui est équivalent à la condition demandée. \square

Dans le cas où l'on donne, en plus des vecteurs ℓ et c , un vecteur δ qui indique le nombre de carreaux noirs dans chaque diagonale, on ne connaît pas d'algorithme polynomial en n qui décide s'il existe une matrice ayant ℓ , c et δ comme projection. On peut montrer, mais la réduction est assez complexe, que ce problème est NP-complet. Ce type de problèmes est en réalité motivé par des questions de reconstruction d'images à partir de projections (radiographies ou RMN en imagerie médicale, cristallographie, etc).

II. Couplages, algorithme glouton et connexité

Le client exige de plus qu'il n'y ait pas de carré noir sur la première diagonale et que la figure formée par les carrés noirs et blancs soit symétrique par rapport à cette diagonale (en particulier il faut que $c = \ell$). Un exemple est donné à la figure 1(b).

1. Montrer que le problème est équivalent à la recherche d'un graphe non orienté à n sommets de degrés respectifs $d_1 \geq d_2 \geq \dots \geq d_n \geq 0$ (le degré d'un sommet est son nombre de voisins). Représenter graphiquement le graphe associé au carrelage de la figure 1(b). Calculer le nombre m d'arêtes d'un tel graphe.

Corrigé: Les matrices 0-1 symétriques de diagonale nulle sont exactement les matrices d'adjacence de graphes. La somme des entrées dans une ligne donne le nombre d'arêtes incidentes au sommet correspondant, i.e. son degré. On peut supposer les d_i décroissants par permutation simultanée des lignes et colonnes. Par définition des degrés des sommets, $2m = \sum_{i=1}^n d_i$. \square

2. Modéliser ce problème par la recherche d'un couplage dans le graphe $G = (X, E)$ avec $X = X_1 \cup X_2$ et $E = E_1 \cup E_2$ où
 - $X_1 = \{v_{i,j} \mid 1 \leq i \neq j \leq n\}$, de sorte que $|X_1| = n(n-1)$,
 - $X_2 = \{w_{i,k} \mid 1 \leq i \leq n, 1 \leq k \leq n-d_i\}$, de sorte que $|X_2| = \sum_i (n-d_i)$,
 - $E_1 = \{\{v_{i,j}, v_{j,i}\} \mid 1 \leq i \neq j \leq n\}$, de sorte que $|E_1| = n(n-1)$,
 - $E_2 = \{\{v_{i,j}, w_{i,k}\} \mid 1 \leq i \neq j \leq n, 1 \leq k \leq n-d_i\}$, de sorte que $|E_2| = \sum_i (n-1)(n-d_i)$.

Corrigé: À un couplage C du graphe G on associe le graphe de sommets $\{1, \dots, n\}$ ayant une arête $\{i, j\}$ pour chaque arête $\{v_{i,j}, v_{j,i}\}$ dans $C \cap E_1$. Si le couplage est parfait, c'est à dire si chaque sommet de G est couplé, alors exactement $n-d_i$ sommets $v_{i,j}$ sont couplés à des sommets $w_{i,k}$; il reste donc exactement d_i sommets $v_{i,j}$ couplés à d'autres sommets, qui ne peuvent être que les $v_{j,i}$ correspondants; le sommet i du graphe associé est alors bien de degré d_i .

Réciproquement étant donné un graphe H satisfaisant on construit un couplage parfait en couplant les paires de sommets $\{v_{i,j}, v_{j,i}\}$ de G associées aux arêtes de H et en complétant le couplage par les arêtes vers E_2 : pour chaque i on dispose bien de $n-d_i$ sommets $v_{i,j}$ libres d'être couplés avec les $w_{i,k}$.

On est donc ramené à chercher un couplage parfait, i.e. de cardinal $\frac{1}{2}n(n-1) + \frac{1}{2}\sum_{i=1}^n (n-d_i)$. Le graphe n'étant pas bipartite il faut a priori faire appel à l'algorithme d'Edmonds dont on a mentionné plusieurs fois l'existence en cours, polynomial mais assez complexe. \square

On cherche une caractérisation des suites de degrés pour lesquelles il existe un graphe.

2. Montrer que si les sommets d'un graphe ont pour degré $d_1 \geq \dots \geq d_n \geq 0$ alors

$$\sum_{i=1}^k d_i \leq k(k-1) + \sum_{i=k+1}^n \min(k, d_i).$$

Corrigé: Il suffit de compter les arêtes qui sont issues des k premiers sommets: au plus $k(k-1)$ sont internes à ces k sommets, au plus k arrivent en chacun des autres sommets. \square

En fait cette condition est suffisante pour que les d_i soient la suite des degrés d'un graphe, mais la preuve étant assez longue on admettra ce résultat.

Le graphe solution trouvé par la réduction à un problème de couplage n'est pas forcément connexe. Quelques rappels : deux sommets x et y d'un graphe H sont *connectés* s'il existe un chemin de x à y dans H ; une *composante connexe* de H est un ensemble maximal de sommets 2 à 2 connecté dans H ; un graphe est *connexe* s'il ne possède qu'une seule composante connexe ; un graphe connexe possède (au moins) un arbre couvrant.

3. À partir de la comparaison du nombre de sommets et du nombre d'arêtes d'un graphe connexe, donner une condition nécessaire sur la somme des d_i pour qu'il puisse exister un graphe solution connexe.

Corrigé: *Il faut qu'il existe un arbre couvrant, donc qu'il y ait au moins $n - 1$ arêtes pour n sommets. On en déduit qu'il faut $\sum_{i=1}^n d_i \geq 2n$. \square*

4. Étant donné un graphe non connexe satisfaisant la condition précédente, proposer une modification locale d'une paire d'arêtes bien choisies qui diminue le nombre de composantes connexes sans changer les degrés des sommets.

En déduire un algorithme pour transformer un graphe non connexe de suite de degré $d_1 \geq \dots \geq d_n$ en un graphe connexe de même degré. Quelle est la complexité de votre algorithme ?

Corrigé: *Si le graphe n'est pas connexe et contient au moins $n - 1$ arêtes alors il existe un cycle. Soit $e = \{x, y\}$ une arête d'un cycle et soit $e' = \{x', y'\}$ une arête d'une autre composante connexe que celle de e . On remplace e et e' par $\{x, x'\}$ et $\{y, y'\}$. Les degrés des sommets ne changent pas mais le nombre de composantes connexes diminue de 1. Chaque étape nécessite de chercher une arête cyclique : pour cela peut utiliser un parcours en profondeur par exemple, pour une complexité en $O(m)$. Il peut être nécessaire de recommencer $m - n$ fois, soit $O(m(m - n))$. \square*

5. Déduire des résultats précédents une condition nécessaire et suffisante sur une suite $d_1 \geq \dots \geq d_n \geq 0$ pour qu'elle soit la suite des degrés d'un graphe connexe.

Corrigé: *Il suffit de mettre ensemble les conditions vues précédemment : $\sum_{i=1}^k d_i \leq k(k - 1) + \sum_{i=k+1}^n \min(k, d_i)$ et $\sum_{i=1}^n d_i = 2n$. \square*

III. Programmation dynamique

Un client désire un carrelage de taille $m \times n$ formé de carrés unité blancs et d'une frise noire, constituée de carreaux rectangulaires de taille $1 \times k$ pour différentes valeurs de k . La frise part du carré $(1, 1)$ et arrive au carré (m, n) et doit progresser de gauche à droite lorsqu'on la regarde tournée de 45° (pas de retour en arrière). Elle doit de plus éviter des obstacles indiqués par une matrice booléenne $P = (p_{i,j})_{1 \leq i, j \leq n}$: $p_{i,j}$ vaut 1 si on peut poser un carreau en position (i, j) , 0 sinon.

1. On suppose dans un premier temps qu'on dispose d'une réserve infinie de chaque type de carreaux noirs : les carreaux de type i sont de taille $1 \times \ell_i$ et de coût c_i , pour $i = 1, \dots, k$. Donner un algorithme polynomial qui propose une frise réalisable de coût minimum. Quel est la complexité de votre algorithme ?

Corrigé: *1ère solution : On construit un graphe ayant les cases libres comme sommets et les arêtes suivantes : pour toute case (p, q) et toute type*

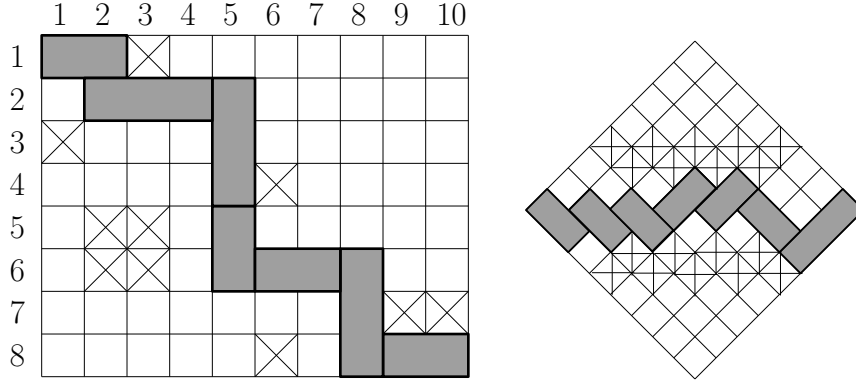


FIG. 2 – (a) une frise de taille 8×10 (b) une frise 8×8 , tournée de 45°

de pièce i , on ajoute des arêtes de poids c_i arrivant en (p, q) à partir des points $(p - \ell_i - 1, q)$, $(p - \ell_i, q - 1)$, $(p - 1, q - \ell_i)$ et $(p, q - \ell_i - 1)$ lorsque les emplacements correspondants sont libres. Puis on cherche un chemin de minimal dans ce graphe.

2ème solution : Par programmation dynamique dans un tableau de taille $(m + 1) \times (n + 1)$: Au départ $t_{0,1} = t_{1,0} = 0$, $t_{p,q} = +\infty$ sinon. Puis on utilise dans 2 boucles imbriquées sur p et q la récurrence

$$t_{p,q} = \min_{i=1,\dots,k} \min \left\{ \begin{array}{ll} t_{p-\ell_i,q} + c_i & \text{si les cases } (p - \ell_i + 1, q) \text{ à } (p, q) \text{ sont libres} \\ t_{p-\ell_i+1,q-1} + c_i & \text{si les cases } (p - \ell_i + 1, q) \text{ à } (p, q) \text{ sont libres} \\ t_{p-1,q-\ell_i+1} + c_i & \text{si les cases } (p, q - \ell_i + 1) \text{ à } (p, q) \text{ sont libres} \\ t_{p,q-\ell_i} + c_i & \text{si les cases } (p, q - \ell_i + 1) \text{ à } (p, q) \text{ sont libres} \\ +\infty & \end{array} \right.$$

Comme d'habitude on trouve le coût de la frise optimale en $t_{n,m}$ et on retrouve la frise optimale en remontant de $t_{n,m}$ vers $t_{1,0}$ ou $t_{0,1}$. La complexité est en $O(nmk)$. \square

2. On suppose maintenant qu'on ne dispose que d'une réserve finie de k carreaux : le i ème carreau est de forme $1 \times \ell_i$ et à un coût c_i . On cherche parmi les frises réalisables avec les réserves disponibles une de coût minimum. Donner pour cela un algorithme polynomial en m , n et k .

Corrigé: Par programmation dynamique sur le tableau $t_{p,q,i}$ contenant le coût de la meilleure frise allant jusqu'en (p, q) en utilisant seulement des pièces prises parmi les i premières. La récurrence de la question précédente s'adapte immédiatement et on remplit le tableau en ajoutant une boucle externe pour $i = 1, \dots, k$. \square

IV. NP-complétude

On reprend tout d'abord le problème de l'exercice précédent dans le cas simplifié où il n'y a pas d'obstacle. Autrement dit on considère le problème FRISE suivant :

Donnée : la taille (m, n) de la grille et une liste $(\ell_1, c_1), \dots, (\ell_k, c_k)$ de k pièces (de forme $1 \times \ell_i$ et de coût c_i);

Problème : Trouver si elle existe une frise joignant $(1, 1)$ à (m, n) de coût minimum.

On remarque alors que les données (n, m et la liste des k pièces disponibles) sont de taille $N = O(k(\log(n + m + C)))$ où C est le coût maximum d'une pièce : on se pose le problème de l'existence d'un algorithme polynomial en N pour résoudre FRISE.

1. Énoncez le problème de décision associé à la recherche d'une frise de coût minimum.

Corrigé: *Existe-t-il une frise de coût inférieur à c ?* \square

2. Montrez que ce problème est NP-complet.

Corrigé: *Réduction triviale à partir de sac-à-dos puisque pour $m = 1$ c'est exactement le problème du sac-à-dos.* \square

Pour montrer la NP-complétude du problème mentionné à la fin de l'exercice 1, une première étape est de montrer la NP-complétude du problème TRIPARTITION suivant :

Donnée : des entiers positifs x_1, x_2, \dots, x_n

Problème : existe-t-il une partition de $\{1, \dots, n\}$ en trois parties X_1, X_2, X_3 telle que $\sum_{j \in X_i} x_j = \frac{1}{3} \sum_{j=1}^n x_j$ pour $i = 1, 2, 3$.

3. Montrez que le problème TRIPARTITION est NP-complet.